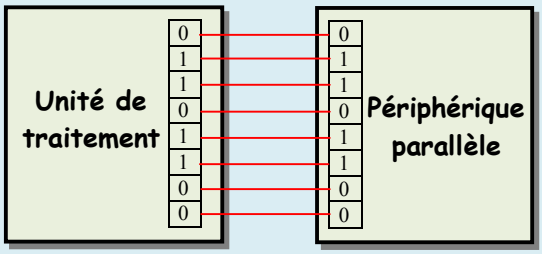
1. Introduction

A l'intérieur, l'unité de traitement d'un système automatisé (API, ...) communique les informations naturellement en parallèle sous un format de 8 bits ou plus. Mais, pour communiquer avec le milieu extérieur, l'unité de traitement communique en général en série, ce qui réduit le câblage.

2. Liaison parallèle

Exemple : Imprimante parallèle

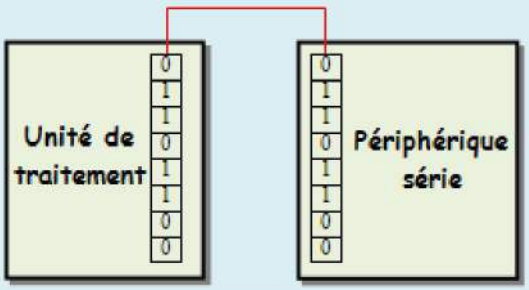
Un texte est un ensemble de caractères, chaque caractère est codé sur un mot de 8 bits (code ASCII). Avec une imprimante parallèle, l'ordinateur envoie le texte sous forme d'une succession de caractères de 8 bits l'un après l'autre. On utilise un connecteur **SUB-D25** côté PC et « **Centronics** » côté imprimante.



3. Liaison série

31. Mise en situation

Lorsque la distance devient grande, la liaison parallèle devient techniquement difficile à réaliser à cause de la longueur du câblage et des parasites de transmission, on utilise alors la liaison série.
Par exemple, un API est généralement programmé avec un ordinateur, le mode de communication dans cette situation est le mode série.



Les 8 bits sont communiqués en série un par un sur un seul fil

Programmation d'API avec PC

Dans une liaison série, on distingue plusieurs procédés et techniques de transmission, dans ce qui suit on s'intéresse à la liaison série asynchrone, en particulier avec les normes **RS232** et **RS485**.

32. Principe de la liaison série asynchrone

321. Format

La liaison série asynchrone est orientée pour une transmission par caractères, ces derniers sont envoyés individuellement et l'intervalle séparant 2 octets est quelconque. Ce mode asynchrone utilise un format où chaque caractère :

Fonction Communiquer

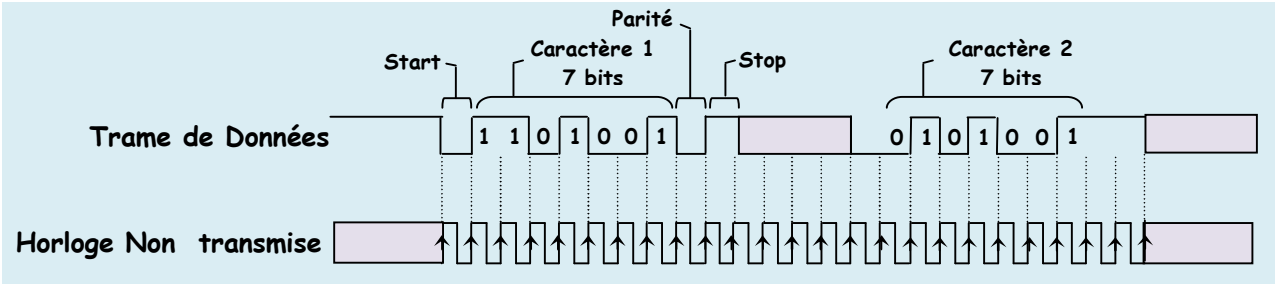
➢ A une longueur de **5** à **8** bits.
➢ Est encadré par des bits délimiteurs :

↳ 1 bit **Start** au début de chaque caractère.
↳ 1 à 2 bits **Stop** à la fin de chaque caractère.

➢ Peut être protégé, contre les parasites de transmission, par un bit de parité optionnel destiné à la détection d'erreurs (suivant la configuration du système). Il est généré à l'émission et testé à la réception. Il existe deux types de parité :

↳ **Parité paire** : si le nombre de bits (donnée + bit parité) au niveau logique 1 est pair.
↳ **Parité impaire** : la parité est dite impaire pour un nombre impair de bits à 1.

L'ensemble (bit Start, Bits données, Bit parité, Bits Stop) est appelé "**trame**" (**frame**). Voici un exemple de trame série asynchrone avec une parité paire, elle représente la transmission de la lettre "**K**" dont le code **ASCII** est $(75)_{10}=(1001011)_2$. On remarque que le bit **LSB** est transmis le 1er et convention de parité paire :



322. Fonctionnement

Le fonctionnement est donc comme suit :

➢ Au repos, la ligne de transmission se trouve dans l'état logique **1**.
➢ Au début de la transmission d'un caractère, on commence par le bit de **Start**, qui dure une période d'horloge.
➢ On enchaîne par les bits du caractère en commencent par le **LSB**.
➢ On termine par le bit de **Stop**.
➢ A la réception de chaque caractère, l'initialisation a lieu par la transition **haut-bas** du bit **Start** qui assure la synchronisation des deux horloges (émetteur et récepteur). Dès la réception du bit **Stop**, il n y a plus de synchronisation.

33. Norme RS232

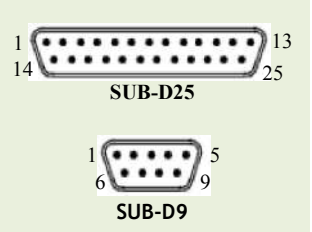
331. Liaison possible

La norme **RS232** définie par l'**EIA** (**E**lectrical **I**ndustry **A**ssociation), correspond à la norme **ISO 2110**. Elle permet une liaison "**point à point**". Il ne peut y avoir que **2** éléments communicants.

332. Caractéristiques

Définition des signaux et connectique

La liaison **RS232** est une interface de tension pour la transmission série aussi bien synchrone qu'asynchrone, utilisée pour les liaisons **point à point**.
Elle est définie pour un connecteur **SUB-D25** ou **SUB-D9**. Elle comporte plusieurs signaux qu'on peut rassembler en deux groupes de fonctions :



➢ Signaux de communication principaux : **Tx** et **Dx**.
➢ Signaux de dialogue "optionnels" : **RTS**, **DTR**, ...

Fonction Communiquer

Longueur de ligne et vitesse

La norme **RS232** est aussi caractérisée par :

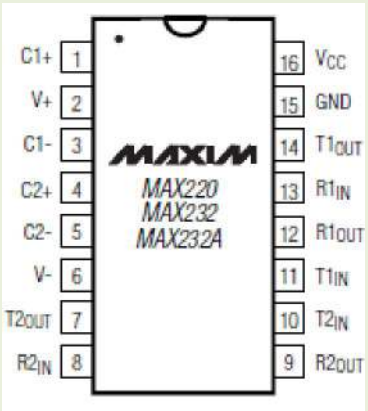
- La longueur maximale du câble qui est d'environ **15** mètres.
- Le débit maximal qui est à présent de **20 Kbits/s**. La norme prévoit ainsi des débits (en bits/s) de 75, 150, 300, 600, 1200, 2400, 4800, 9600 et 19200.

Niveaux de tension

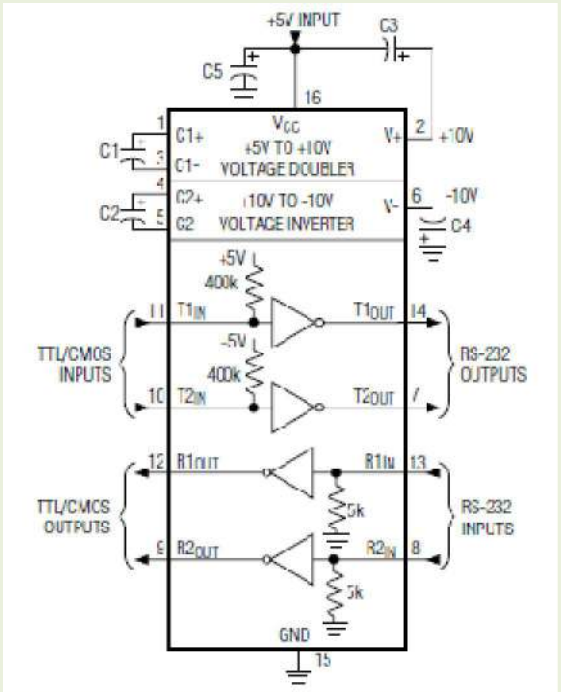
Pour la résistance au bruit, les niveaux de tension de la **RS232** sont plus grands que ceux de la **TTL/CMOS**. L'équivalence avec les niveaux logiques sont décrits par le tableau suivant :

Niveau logique	Polarité	Intervalle de niveau électrique	Typique
0	Haute	De +5 V à +15 V	+12 V
1	Basse	De -5 V à -15 V	-12 V

Les circuits logiques à l'origine de la transmission sont compatibles **TTL/CMOS**, il faut alors des circuits d'adaptation à la norme **RS232**. On cite à titre d'exemple le circuit **MAX232**.
Le convertisseur **MAX232** est un composant créé par le constructeur **MAXIM** que l'on trouve sous d'autres références chez d'autres fabricants. Il sert d'interface entre une liaison série **TTL** et une liaison série **RS232** et ce avec une simple alimentation +5 V.



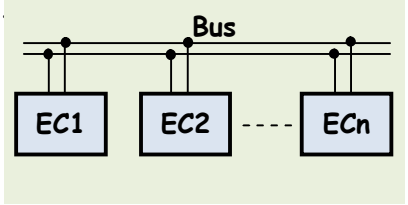
CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



34. Norme RS485

341. Liaison possible

La norme **RS485** est définie par l'**EIA** correspondant à la norme **ISO 2593**. Elle permet une liaison "**multipoints**", c'est à dire, entre plusieurs Eléments Communicants (**EC1**, **EC2**,..., **ECn**). Dans ce cas, il faut qu'il n'y ait qu'un seul élément (**1/n**) qui émet dans le Bus, le reste des éléments reçoit (il est à l'écoute). Dans cette liaison multipoints, on désigne la ligne de transmission de "**Bus**".

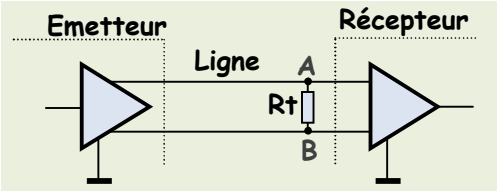


Fonction Communiquer

Il est évident que pour qu'une communication soit possible, il faut que chaque EC dispose d'une interface pour le Bus, plus précisément la possibilité de se mettre en haute impédance, ainsi, on évite les conflits de Bus, cas où 2 EC émettent sur le Bus.

342. Caractéristiques

Définition des signaux et connectique



La norme **RS485** est surtout utilisée dans les réseaux locaux industriels aussi bien point à point que multipoints. Elle utilise un support de transmission différentiel : le signal est transformé en deux signaux complémentaires (**A** et **B**), ce qui assure une résistance aux parasites industriels et augmente la longueur maximale de la ligne. La norme ne précise pas de connecteur spécifique.
Rt : Résistance de terminaison de câble. Elle boucle la ligne sur son impédance caractéristique et minimise le bruit pour une meilleure transmission.

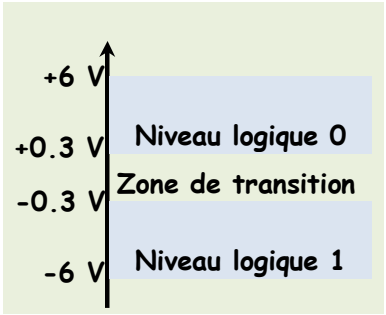
Longueur de ligne et vitesse

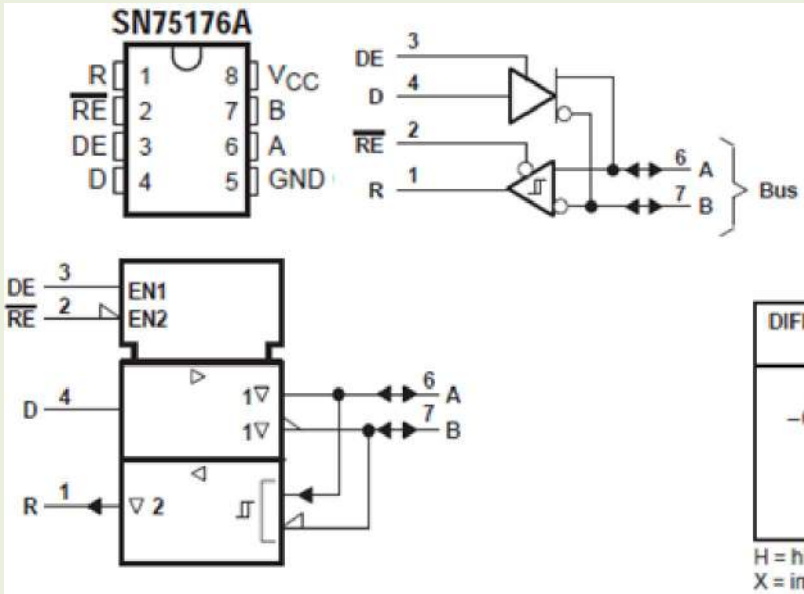
La norme **RS485** est aussi caractérisée par :

- La longueur maximale du câble qui est de **1000** m.
- Le débit maximal qui est à présent de **100 Kbits/s**. La norme prévoit ainsi des débits (en bits/s) de 75, 150, 300, 600, 1200, ...

Niveaux de tension

L'équivalence avec les niveaux logiques sont décrits par le graphique ci-contre. Les circuits logiques à l'origine de la transmission sont compatibles **TTL/CMOS**, il faut alors des circuits d'adaptation à la norme **RS485**, pour convertir une tension bipolaire en une tension différentielle. Parmi les circuits les plus utilisés à cette fin, on trouve l'**AD485** et le **SN75176**.





DRIVER

INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

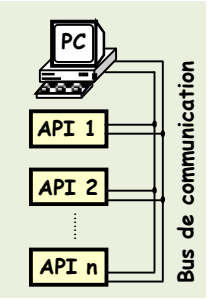
DIFFERENTIAL INPUTS A – B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	?

H = high level, L = low level, ? = indeterminate,
X = irrelevant, Z = high impedance (off)

Fonction Communiquer

343. Notion de protocole

La figure ci-contre montre une architecture typique dans un environnement industriel, avec la norme **RS485**. Les **n API** de commande et le PC de supervision sont reliés donc en réseau, ils peuvent donc échanger des informations. Cela a pour avantage :



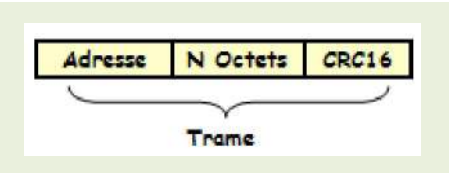
- Une programmation structurée, par exemple, pour un système automatisé à plusieurs postes, on réserve à chaque poste un **API**.
- Un gain en câblage dans l'application.
- Une facilité de maintenance.

Il va sans dire que le **Bus** est partagé et doit donc connaître un "**arbitrage**", en effet, d'après la structure physique du réseau (**norme RS485**), il ne peut y avoir qu'un seul élément qui émet sur le bus, le reste écoute, d'où la nécessité de s'accorder sur des règles de communication, qu'on appelle **protocole**.
Un protocole doit donc résoudre les problèmes liés aux questions suivantes :

- Quel est le format de la trame ?
- A qui s'adresse la trame sur le bus ?

A titre d'exemple, on donne le principe d'un protocole largement diffusé dans ce domaine, il s'agit de "**ModBus**" de **Modicon**. C'est une structure "**Maître/Esclave**" (Master/Slave). Dans ce protocole, il y a un **seul maître** (exemple le **PC**) et **n esclaves** (exemple les **API**). Dans le cas de la figure ci-dessous :

- Le PC est le maître, les n API sont les esclaves.
- Chacun des API a une adresse.
- Le PC envoie une demande à un API et attend une réponse.
- L'API interrogé répond à la demande du PC.
- Chaque demande du maître ou réponse d'esclave est un ensemble d'octets (**trame**) qui a le format ci-contre :



Adresse : 1 octet représentant l'adresse de l'esclave.
N octets : Ces **N octets** représentent l'objet de la demande du maître ou de la réponse d'un esclave.
CRC16 : **2 octets** de détection d'erreur, calculé suivant un algorithme précis, d'après les octets (Adresse + N Octets).